5Б

УДК 519.854.6

А.С. Бондаренко, И.В. Козин
Запорожский национальный университет, г. Запорожье, Украина
buenasdiaz@gmail.com, ainc@ukrpost.net

Эволюционный и фрагментарный подходы к задаче о равномерной нагрузке

В работе представлены результаты исследования задачи о равномерной нагрузке. Подобная задача возникает, в частности, при моделировании структурных элементов учебного процесса в высшем учебном заведении. Проблема заключается в распределении учебных дисциплин во времени таким образом, чтобы максимум нагрузки на студента был минимален. Предложены эволюционный подход для поиска оптимального решения и фрагментарный подход для построения допустимого решения задачи о равномерной нагрузке. Предложенные методы протестированы на наборе случайных задач.

Введение

Целью данной работы является исследование задачи о построении расписания, удовлетворяющего критерию минимума максимальной нагрузки. Исследуемая задача является подзадачей для общей задачи построения расписания занятий в высшем учебном заведении. Различные постановки этой задачи рассматривались в русскоязычной литературе [1-3]. Известен в англоязычной литературе обзор [4]. Также из англоязычной литературы могут быть выделены работы [5-7]. В [8] предложена единая модель для задачи об университетском расписании предметов, основанном на учебной программе (Curriculum-based course timetabling problem). Общая задача является NP-полной [9], [10] в большинстве своих постановок. Исходя из этого для решения таких задач целесообразно рассматривать эвристические и метаэвристические алгоритмы. В [11] предлагается обзор различных метаэвристик для задач об университетском расписании (University timetabling problems). Классом таких метаэвристик является класс эволюционных алгоритмов, схема функционирования которых описывается в настоящей работе. В работе также предлагаются новые подходы к поиску приближенного решения для исследуемой задачи.

Постановка задачи. Задача о равномерной нагрузке рассматривается далее как задача теории расписаний. Общая модель теории расписаний включает три конечных множества: множество машин $\mathrm{M} = \{M_1,...,M_{|\mathrm{M}|}\}$, множество работ $\mathrm{J} = \{J_1,...,J_{|\mathrm{J}|}\}$ и поставленное во взаимное соответствие каждой работе множество операций $O^j = \{o_1^j,...,o_{r_j}^j\}$ [12]. В рассматриваемой задаче множество машин M будет иметь мощность 1. Будем полагать, что машина может выполнять одновременно более одной операции (единичным интервалом времени является одна неделя). Длина расписания R задана в условии, ею может быть длительность семестра или учебного года. Для каждой работы $J_J \in \mathrm{J}$ на множестве её операций O^j задан линейный порядок. Работы $J_J \in \mathrm{J}$ предполагаются непрерывными, то есть для любых двух операций o_k^j и o_{k+1}^j разность времени начала s_{k+1}^j операции o_{k+1}^j и времени завершения f_k^j операции o_k^j равна нулю $s_{k+1}^j - f_k^j = 0$.

Работа может быть начата в любой момент не позже $R - l_J + 1$, где l_J – время выполнения работы J_J. Следовательно, число возможных решений для данной задачи равно $N = \prod_j (R - l_J + 1)$. Под расписанием будем понимать множество моментов начала выполнения работ $S = \{s^1,..., s^{|J|}\}$. В качестве критерия равномерности расписания будем рассматривать максимум объема учебной нагрузки в неделю на одного студента. Критерий может быть вычислен по формуле $L = \max_i (L_c, L_i)$, где L_c – текущий максимум нагрузки, L_i – максимум нагрузки за период *i*. L_i вычисляется как $L_i = L_{i,j-1} + p_{i,j}$, где $L_{i,j-1}$ – нагрузка в период *i*, полученная от *j* – 1 работ, $p_{i,j}$ – время выполнения работы J_J в период *i*.

Далее предлагается ещё более сузить класс рассматриваемых задач с целью приближения к практической постановке. На протяжении семестра занятия по данной дисциплине могут повторяться с определённой периодичностью, например, одна пара в две недели, тогда при условии множество, что запланировано шесть занятий, длина (время выполнения) работы равна $l_j = p_j \cdot n_J - k_j$, где p_j – длина периода работы J_J, n_J – число повторений периода, k_j – число крайних нулей, то есть таких, которые стоят в начале или в конце работы. Такая работа может быть представлена следующим вектором $(2,0,2,0,2,0,2,0,2,0)$, если взять период $(2,0)$. Работы, в которых последовательности операций с одинаковым временем выполнения и одинаковой позицией в последовательности периодично повторяются, в дальнейшем будут называться периодическими работами. Каждая такая работа может быть задана указанием периода и числа его повторений. Поскольку при построении решения крайние нули не учитываются, то, например, работа с периодом $(0,2)$ будет эквивалентна работе с периодом $(2,0)$ при одинаковом числе повторений периодов в этих работах.

1. Описание эволюционной модели

Одним из современных направлений поиска оптимальных решений дискретных оптимизационных задач является подход основанных на использовании генетических (или более обще – эволюционных) алгоритмов.

Этот подход приводит к понятию эволюционной модели, которая состоит из следующих элементов:

– базовое множество решений *X*, на котором ищется оптимальное решение;

– оператор построения начальной популяции: оператор, который позволяет выделить на множестве X его подмножество $X_0 \subseteq X$;

– оператор кроссовера $K : X \times X \to X$, позволяющий по двум перестановкам-родителям построить новую перестановку-потомок из множества *X*;

– оператор мутации $M : X \to X$;

– критерий отбора – алгоритм, позволяющий сравнивать по качеству решения в рамках заданной популяции;

– оператор селекции, позволяющий выделять множество пар в популяции для выполнения кроссовера;

– оператор отбора, позволяющий строить новые популяции из множеств родителей и потомков;

– условие остановки, которое определяет условие остановки эволюции.

Эволюционный алгоритм поиска оптимального решения в эволюционной модели работает следующим образом. На каждом шаге с номером k формируется некоторое множество $X_k \subseteq X$ – текущая популяция. Для каждого элемента текущей популяции вычисляется значение критерия отбора. На начальном шаге X_0 – это начальная популяция, построенная с помощью оператора построения начальной популяции.

Далее с помощью оператора селекции выбирается множество пар текущей популяции для кроссовера. К каждой паре из выбранного множества пар применяются операторы кроссовера и мутации, которые позволяют получить множество потомков. Для каждого потомка также вычисляется значение критерия. С помощью оператора отбора на основе вычисленных значений критерия отбора из объединения текущей популяции и множества потомков формируется новая текущая популяция X_{k+1}, которая содержит элементы с наибольшими значениями критерия отбора.

Работа алгоритма заканчивается при выполнении условия остановки.

2. Представления решений для эволюционного алгоритма

Использовались два представления решения задачи о равномерной нагрузке: векторами смещений и перестановками.

Хромосома в виде вектора смещений $(i_1,...,i_l,...,i_n)$, $0 \le i_l \le R - l_l$, где R – длина расписания, l_l – время выполнения работы J_l, может содержать такие i_j и i_k, $j \ne k$, что $i_j = i_k$. Каждое i_l представляет разность между начальным моментом времени в расписании и моментом времени, в который начинается работа J_l.

Скрещивание хромосом осуществляется при помощи функции $f(i_k^1, i_k^2)$, которая ставит в соответствие элементам вектора смещений родительских хромосом i_k^1, i_k^2, занимающим k-ю позицию в каждой из них, некоторое i_k^2. Такой функцией могут быть $\max(i_k^1, i_k^2)$, $\min(i_k^1, i_k^2)$ или $(i_k^2 - i_k^1) \cdot rnd + i_k^1$, где rnd – генератор случайных чисел на интервале $(0;1]$.

Оператор мутации заключается в случайной генерации номера позиции в векторе смещений и значения времени начала выполнения работы, соответствующей данной позиции.

При представлении решения перестановкой элемент π_j перестановки π представляет номер работы. Работы ставятся в расписание в порядке их вхождения в перестановку, при этом каждая работа ставится в расписание таким образом, чтобы минимизировать максимум нагрузки машины. Таким образом, на каждом шаге принимается локально-оптимальное решение. Оператор кроссовера работает таким образом, что получает на вход две родительские хромосомы-перестановки π^1 и π^2, затем просматриваются элементы на первых позициях в этих хромосомах и в зависимости от значения i-го элемента некоторого бинарного вектора $(b_1,...,b_i,...,b_n)$ i-й ген хромосомы-потомка π^2 наследует значение гена из π^1, если $b_i = 0$, или из π^2, если $b_i = 1$. Значение, унаследованное π^2, удаляется из π^1 и π^2.

Описанный кроссовер сохраняет свойство предшествования, то есть если ген со значением a предшествует гену со значением b в обеих родительских хромосомах, то данное отношение сохранится и в хромосоме-потомке.

Оператор мутации для перестановок генерирует две случайные позиции в перестановке и меняет их значения местами.

3. Фрагментарная структура

В соответствии с [13] фрагментарной структурой будем называть тройку (X,Y,R), где X – конечное множество элементарных фрагментов, Y – семейство подмножеств множества X, R – условие присоединения, т.е. правило, позволяющее ответить на вопрос, является ли объединение множеств из X элементом множества Y.

Задача о равномерной нагрузке может быть сформулирована как задача с фрагментарной структурой. Фрагментами в этой задаче являются любые упорядоченные последовательности работ, соответственно в роли элементарных фрагментов выступают отдельные работы. Условие присоединения – это допустимое время начала присоединяемой работы. Максимальный по включению фрагмент соответствует допустимому решению рассматриваемой оптимизационной задачи. Таким образом, допустимое решение можно рассматривать как некоторую фиксированную перестановку работ π .

В [13], [14] показано, как для отыскания допустимых решений подобных комбинаторных задач может быть использована фрагментарная структура задачи. Для построения допустимого решения применяется фрагментарный алгоритм F. В общем случае, на вход фрагментарного алгоритма F подаётся последовательность $(i_1, i_2, ..., i_n)$ номеров элементарных фрагментов, которая просматривается в порядке нумерации, и фрагмент-решение S, который в начальный момент пуст. На каждом шаге просмотра алгоритма ищется первый элемент в упорядоченной последовательности элементарных фрагментов, для которого выполняется условие присоединения. Это фрагмент добавляется к S. Процедура повторяется до тех пор, пока не будет построен максимальный по включению фрагмент.

4. Результаты тестирования

Предложенные алгоритмы были реализованы на языке Visual Basic for Applications и выполнены на процессоре Intel Core 2 Duo с тактовой частотой ядра 2 ГГц. Было сгенерировано 120 случайных задач. В этих задачах использовался период для работ вида (2,0). Для генерации использовался генератор задач, взятый из [15] с теми же начальными условиями.

Результаты работы алгоритма сравнивались с работой метода случайного поиска. Параметры, которыми задавались индивидуальные задачи, были следующими:
– R – длина расписания (10, 20, 50, 100);
– n – число работ (10, 20, 50, 100);
– 1, R – интервал, из которого случайно выбирается длина работы.

Таким образом, получено 12 групп индивидуальных задач размерностью R на n по 10 индивидуальных задач в каждой группе. Результаты применения названных выше подходов к решению рассматриваемой задачи приводятся в табл. 1.

Для каждой задачи каждый алгоритм запускался не менее 30 раз. Для полученных таким образом значений целевой функции было вычислено среднее. Значения целевой функции, которые приведены в табл. 1, представляют собой средние для каждой группы задач одинаковой размерности, вычисленные от средних для каждой индивидуальной задачи.

В наборе протестированных алгоритмов выделяются такие свойства: локальная оптимальность решения (ЛОР), наличие глобального поиска (в данном случае

эволюции). Эволюционный метод и случайный поиск и фрагментарный алгоритм с решениями, заданными на перестановках ЭА(пр), СП(пр), ФА, включают свойство ЛОР. В эволюционном алгоритме и случайном поиске с решениями, определенными векторами смещений ЭА(вс), СП(вс), ЛОР отсутствует. Для сравнения был отобран ФА с упорядочиванием по убыванию времени выполнения работ, поскольку именно с таким упорядочиванием ФА показал наилучшие результаты.

Таблица 1 – Результаты применения различных подходов

Алгоритм / Размерность	СП (вс)	ЭА(вс)	СП (пр)	ЭА(пр)	ФА
10×10	8	8	8	7	7
10×20	15	14	13	13	13
10×50	40	36	33	32	32
10×100	80	72	66	63	62
20×20	16	15	13	13	13
20×50	38	35	30	30	29
20×100	76	70	62	61	58
50×20	16	15	13	12	12
50×50	38	35	30	30	28
50×100	76	71	62	61	57
100×50	38	35	31	30	29
100×100	77	72	62	62	57

Результаты эксперимента показали, что изменение свойств в алгоритме СП(вс) в следующем порядке ведет к улучшению качества решения:

1. Эволюция.
2. ЛОР.
3. Эволюция и ЛОР.
4. ЛОР и упорядочивание работ по убыванию их времени выполнения.

Таким образом, можно сделать вывод о том, что эволюция и ЛОР улучшают по отдельности качество решения по сравнению со случайным поиском без ЛОР. При добавлении к ЛОР упорядочивания работ по убыванию их времени выполнения качество результатов повышается.

Заключение

В данной статье предложены два новых подхода к решению задачи о равномерной нагрузке. Получено представление решений в эволюционной и фрагментарной моделях для задачи теории расписаний. Выполнено тестирование предложенных алгоритмов на случайном наборе задач.

Результаты тестирования, которые приведены в табл. 1, показывают перспективность подхода на основе эволюционной и фрагментарной моделей для поиска приближенных решений задач теории расписаний.

Литература

1. Клеванский Н.Н. Метод формирования расписания занятий университета / Н.Н. Клеванский, С.С. Кашин // Образовательные технологии : Научно-технический журнал. – 2006. – № 2. – С. 83-87.

2. Кулеш М.А Математическая модель оптимального учебного расписания / М.А. Кулеш, В.Г. Павленко, О.И. Скульский, В.Ю. Столбов // Материалы Всероссийской научно-практической конференции «Региональные проблемы информатизации образования». Часть 1. – Пермь : Изд-во Пермского регионального института педагогических информационных технологий, 1999. – С. 240-245.
3. Ускач А.Ф. Динамическая пространственная модель автоматизированной системы формирования расписаний / А.Ф. Ускач // Труды Одесского политехнического университета. – 2006. – № 2. – С. 1-4.
4. Bardadym V.A. Computer-aided school and university timetabling: The new wave / V.A. Bardadym // Proceedings of the First International Conference on Practice and Theory of Automated Timetabling (PATAT 1995), LNCS. – Vol. 1153. – Springer, 1996. – P. 22-45.
5. Chiarandini M. An effective hybrid approach for the university course timetabling problem / M. Chiarandini, K. Socha, M. Birattari, O. Rossi-Doria // Journal of Scheduling. – 2006. – Vol. 9, № 5. – P. 403-432.
6. Kostuch P. The university course timetabling problem with a three-phase approach / P. Kostuch // LNCS, Vol. 3616. – Springer, 2005. – 109 p.
7. Abdullah S. A hybrid evolutionary approach to the university course timetabling problem / S. Abdullah, E.K. Burke, B. McCollum // Proceedings of IEEE Congress on Evolutionary Computation. – 2007. – P. 1764-1768.
8. De Cesco F. Benchmarking Curriculum-Based Course Timetabling: Formulations, Data Formats, Instances, Validation, and Results / [Электронный ресурс] F. De Cesco, L. Di Gaspero, A. Schaerf // Proceedings of Practice and Theory of Automated Timetabling to appear. – 2008. – Режим доступа : http://w1.cirrelt.ca/~patat2008/PATAT_7_PROCEEDINGS /Papers/Schaerf-WC1a.pdf
9. Even S. On the complexity of timetable and multicommodity flow problems / S. Even, A. Itai, A. Shamir // SIAM Journal of Computation. – 1976. – Vol. 5, № 4. – P. 691-703.
10. Cooper T.B. The complexity of timetable construction problems / T.B. Cooper, J.H. Kingston // Proceedings of the First International Conference on Practice and Theory of Automated Timetabling (PATAT-95), LNCS. – Springer, 1996. – Vol. 1153. – P. 283-295.
11. Lewis R. A survey of metaheuristic-based techniques for university timetabling problems / R.A. Lewis // OR Spectrum. – 2008. – Vol. 30, № 1. – P. 167-190.
12. Севастьянов С.В. Геометрические методы и эффективные алгоритмы в теории расписаний : дисс. на соискание уч. степени доктора ф.-м. наук / Севастьянов Сергей Васильевич. – Новосибирск, 2000. – 280 с.
13. Козин И.В. Фрагментарные алгоритмы в системах поддержки принятия решений // Питання прикладної математики і математичного моделювання : зб. наукових праць. – ДНУ : Дніпропетровськ, 2006. – С. 131-137.
14. Козин И.В. Фрагментарный алгоритм для задачи симметричного размещения / И.В. Козин // Радиоэлектроника, информатика, управление. – 2005. – № 1. – С. 76-83.
15. Taillard E. Benchmarks for basic scheduling problems / E. Taillard // European Journal of Operational Research. – 1993. – Vol. 64, № 2. – P. 278-285.

О.С. Бондаренко, І.В. Козін

Еволюційний та фрагментарний підходи до задачі про рівномірне навантаження
У роботі представлені результати дослідження задачі про рівномірне навантаження. Подібна задача виникає, зокрема, при моделюванні структурних елементів навчального процесу у вищому навчальному закладі. Проблема полягає у розподілі навчальних дисциплін у часі в такий спосіб, щоб максимальне навантаження на студента було мінімальним. Запропоновано еволюційний підхід для пошуку оптимальних розв'язків та фрагментарний підхід до побудови допустимих розв'язків. Запропоновані методи протестовані на наборі випадкових індивідуальних задач.

O.S. Bondarenko, I.V. Kozin

In the paper the results of a study of the uniform loading problem are presented. A similar problem arises, in particular, when modeling structural components of a learning process in a university. The problem consists of allocating of courses in time so that student's maximal loading was minimized. An evolutionary approach for an optimal solution search and fragmentary approach for a feasible solution construction are proposed. The proposed methods are tested on instances generated at random.

Статья поступила в редакцию 08.07.2009.